WRDC-TR-89-1149

EVENT-TRAIN RESTORATION VIA BACKPROPAGATION NEURAL NETWORKS

Peter G. Raeth, Capt, USAF
ESM Technology Group
Electronic Warfare Division

December 1989

Interim Report for Period January 1989 - July 1989

Approved for Public Release; Distribution Unlimited

AVIONICS LABORATORY
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

[illegible]
PETER G. RAETH, Capt, USAF
ESM Technology Group, EW Division
Avionics Laboratory

[illegible]
PAUL S. HADORN, PhD, Chief
Passive ECM Branch, EW Division
Avionics Laboratory

FOR THE COMMANDER

[illegible]
JOHN E. TEHAN, Chief
Electronic Warfare Division
Avionics Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) WRDC-TR-89-1149	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Wright Research and Development Center Avionics Lab / 6b. OFFICE SYMBOL (If applicable) WRDC/AAWP-1	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) WPAFB OH 45433-6543	7b. ADDRESS (City, State, and ZIP Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION N/A / 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State, and ZIP Code) N/A	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62204F PROJECT NO. 7633 TASK NO 11 WORK UNIT ACCESSION NO. 13

11. TITLE (Include Security Classification)

Event-Train Restoration Via Backpropagation Neural Networks

12. PERSONAL AUTHOR(S)

Captain Peter G.Raeth

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM Jan 89 TO Jul 89	1989 December	38

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Neural Networks, Connectionism, Parallel Programming, Parallel Architectures, Backpropagation, Electronic Warfare, Signal Identification, Threa Identification

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See Back

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Captain Peter G. Raeth	22b. TELEPHONE (Include Area Code) 513-255-7854 / 22c. OFFICE SYMBOL WRDC/AAWP-1

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Block 19:

The purpose of this project is to investigate backpropagation neural networks for specific applications in passive electronic warfare (EW) involving restoration of deinterleaved event trains to their original broadcast form. This is different from traditional bit-error detection/correction which relies on a prior knowledge of what the original bit stream looked like. In electronic warfare it is unlikely that such prior knowledge will be avaliable.

Three major questions concerning neural networks of all types are: what training equation to use, what values to select for the parameters for these equations, and what is an appropriate network configuration. The authors of the aforementioned questions make some rather positive claims for their answers. Not much in the way of independent validation of these claims is typically done. This project will concentrate first on a review of relevant papers by various authors. Then one or more training techniques will be selected based on the clarity of the write-ups and the reasonableness of their approaches. A study will then be performed on the training techniques to show how well they perform against a standard benchmark, the two-dimensional XOR problem. This involves having the network learn to recognize the XOR binary table. The study will take each author's training equation and vary training equation parameter values and the network configuration. From this variation, convergence maps can be drawn. Convergence maps are a network performance display technique being developed under this project. The advantage of the convergence map is a clear indication of the sensitivity of the training equation to variations in training equation parameter values and network configuration. The convergence map shows how to pick parameter values and network configurations such that the network's ability to learn is not sensitive to small changes in the values and configurations. Once convergence maps have been drawn for different training equations; parameter values and network configurations will be chosen from a section of the map which shows quick convergence and a flat surface. Such a portion of the map will illustrate a range of convergence times which are not sensitive to parameter and configuration selection. The network, using the selected equations, parameters, and configurations, will then be used in an attempt to add missing events and remove extraneous events in a deinterleaved event train.

The results of this research can be applied to three major problem areas: 1) pulse-train restoration, 2) communications signal compression, and 3) data compression. Pulse-Train Restoration: This is an important concern in threat alert. When a radio frequency (RF) signal is deinterleaved, the pulses thus obtained are assigned to various pulse trains according to an algorithm which determines how many pulse emitters are evident in the non-deinterleaved pulse stream. It is not possible for the deinterleaving process to be perfect. Sometimes it will happen that a pulse is added to a train that it does not belong to. At other times it will happen that a pulse is lost completely. The imperfect pulse trains thus generated lead to imperfect threat alert. Neural networks can ease this problem by generalizing the incoming pulse trains against known pulse trains and restoring them to their original form. Communications Signal Compression: An important concern is the future need to process communication signals which are not now practical to try to compress and transmit due to huge data volumes. The basic idea is to not transmit binary data that can be predicted or restored at the receiving end. This problem is related to data compression discussed next. Data Compression: A serious problem which exists in all weapons and communications systems is that of processing limitations. By processing we mean not only the work done by the CPU(s) but also the work done by the sensors and the data bus. Data density within a system drives the processing problem.

Preface

The purpose of this project is to investigate backpropagation neural networks for specific applications in passive electronic warfare (EW) involving restoration of deinterleaved event trains to their original broadcast form. This is different from traditional bit-error detection/correction which relies on a prior knowledge of what the original bit stream looked like. In electronic warfare it is unlikely that such prior knowledge will be avaliable.

Backpropagation neural networks are currently the most popular for engineering applications. Given the amount of work being done on this network type, it is likely that the literature will support the investigation intended by this project. The IEEE Neural Network Conference Proceedings are among those which contain a considerable number of papers on the training of backpropagation neural networks. The Air Force Institute of Technology (AFIT) has published theses on this subject as have a number of other universities. DTIC (Defense Technical Information Center) holdings contain reports on backpropagation implementations published by numerous researchers.

Three major questions concerning neural networks of all types are: what training equation to use, what values to select for the parameters for these equations, and what is an appropriate network configuration. The authors of the aforementioned questions make some rather positive claims for their answers. Not much in the way of independent validation of these claims is typically done. This project will concentrate first on a review of relevant papers by various authors. Then one or more training techniques will be selected based on the clarity of the write-ups and the reasonableness of their approaches. A study will then be performed on the training techniques to show how well they perform against a standard benchmark, the two-dimensional XOR problem. This involves having the network learn to recognize the XOR binary table. The study will take each author's training equation and vary training equation parameter values and the network configuration. From this variation, convergence maps can be drawn. Convergence maps are a network performance display technique being developed under this project. An example three-dimensional map is shown in Atch 1. The advantage of the convergence map is a clear indication of the sensitivity of the training equation to variations in training equation parameter values and network configuration. The convergence map shows how to pick parameter values and network configurations such that the network's ability to learn is not sensitive to small changes in the values and configurations. Once convergence maps have been drawn for different training equations; parameter values and network configurations will be chosen from a section of the map which shows quick convergence and a flat surface. Such a portion of the map will illustrate a range of convergence times which are not sensitive to parameter and configuration selection. The network, using the selected equations, parameters, and configurations, will then be used in an attempt to add missing events and remove extraneous events in a deinterleaved event train.

The results of this research can be applied to three major problem areas: 1) pulse-train restoration, 2) communications signal compression, and 3) data compression. Pulse-Train Restoration: This is an important concern in RF threat alert. When a radio frequency (RF) signal is deinterleaved, the pulses thus obtained are assigned to various pulse trains according to an algorithm which determines how many pulse emitters are evident in the non-deinterleaved pulse stream. It is not possible for the deinterleaving process to be perfect. Sometimes it will happen that a pulse is added to a train that it does not belong to. At other times it will happen that a pulse is lost completely. The imperfect pulse trains thus generated lead to imperfect threat alert. Neural networks can ease this problem by generalizing the incoming pulse trains against known pulse trains and restoring them to their original form. Communications Signal Compression: An important concern is the future need to process communication signals which are not now practical to try to compress and transmit due to huge data volumes. The basic idea is to not transmit binary data that can be predicted or restored at the receiving end. This problem is related to data compression discussed next. Data Compression: A serious problem which exists in all weapons and communications systems is that of processing limitations. By processing we mean not only the work done by the CPU(s) but also the work done by the sensors and the data bus. Data density within a system drives the processing problem. One way to limit the amount of data flowing on the bus is to not send data which can be predicted or restored at the receiving end. Given the nature of neural network technology, the need to restore the data stream to its original form does not necessarily add to the load on the CPU. The data compression problem is similar to the bit compression problem except that bit compression works with binary values while data compression works with analog values. Prediction of analog values is likely to be considerably more difficult.

Table of Contents

Introduction to Experiments

In this first set of experiments we developed Lippmann's traditional backpropagation neural network model with a modification by Klimasauskas. This development was also assisted by Gustafson's notes. The modification by Klimasauskas involved an exception to Lippmann's and Gustafson's specification in that this project's model uses a positive bias term instead of a negative bias term. For the cases tried in this project so far, the negative bias term prevented the network from converging (learning the intended mappings.) It should be recognized, however, that Lutey notes cases where the negative bias term is required for convergence. It is possible, therefore, that the negative bias term will be tried again later in the project.

As a means of getting started, we experimented with Gustafson's fairly simple representation of Lippmann's backpropagation model. This was an effort to learn the fundamentals of the model. We then moved on to implementing and validating a generic three-layer model. All these start up efforts are documented in Raeth's report of Jan 89.

Once the generic three-layer model was completed, a four-layer model was generated and tested. Using these two models, runs were made to generate 2- and 3-dimensional convergence maps based on the 2-dimensional XOR problem. This involved training the network to map the XOR inputs to the desired outputs. Variations were made of parameter values (gain, momentum, and distribution) and network configuration (interlayer connections and number of nodes in the hidden layer). Convergence maps were drawn to show the ability of the network to learn the XOR input/output mappings. The results of these experiments are described in the next chapter. Generally, there are some rather dramatic variations in the network's ability to converge on (learn) the desired mapping, depending on how one picked parameter values and network configurations. However, the results show clearly how, for this class of mapping, to pick values and configurations that are not sensitive to minor variation in parameter values or network configuration. This will benefit the project when we try the more complex mappings required to restore pulse trains.

In looking over the experiment's results, the reader may be moved to ask why certain other experiments were not tried. The answer is, in most cases, that the computing facilities available to this phase of the project were inadequate. We were on a DEC MicroVax III computer under Ultrix using Pascal. Some runs presented here took a calendar-month to turn around in background mode. Later phases of this project will be on the Cray computer under COS using Pascal. Another reason is that this phase of the project was concentrated on developing the general idea of convergence maps and getting some experience with them. By the time all the experiments documented here were finished, we had plenty of experience with convergence maps and were ready to move on to a metric that would be more relevant to our applications. Thus, for the next phase of the project, the metric will be changed from XOR to random-bit-stream replacement.

Results of Experiments

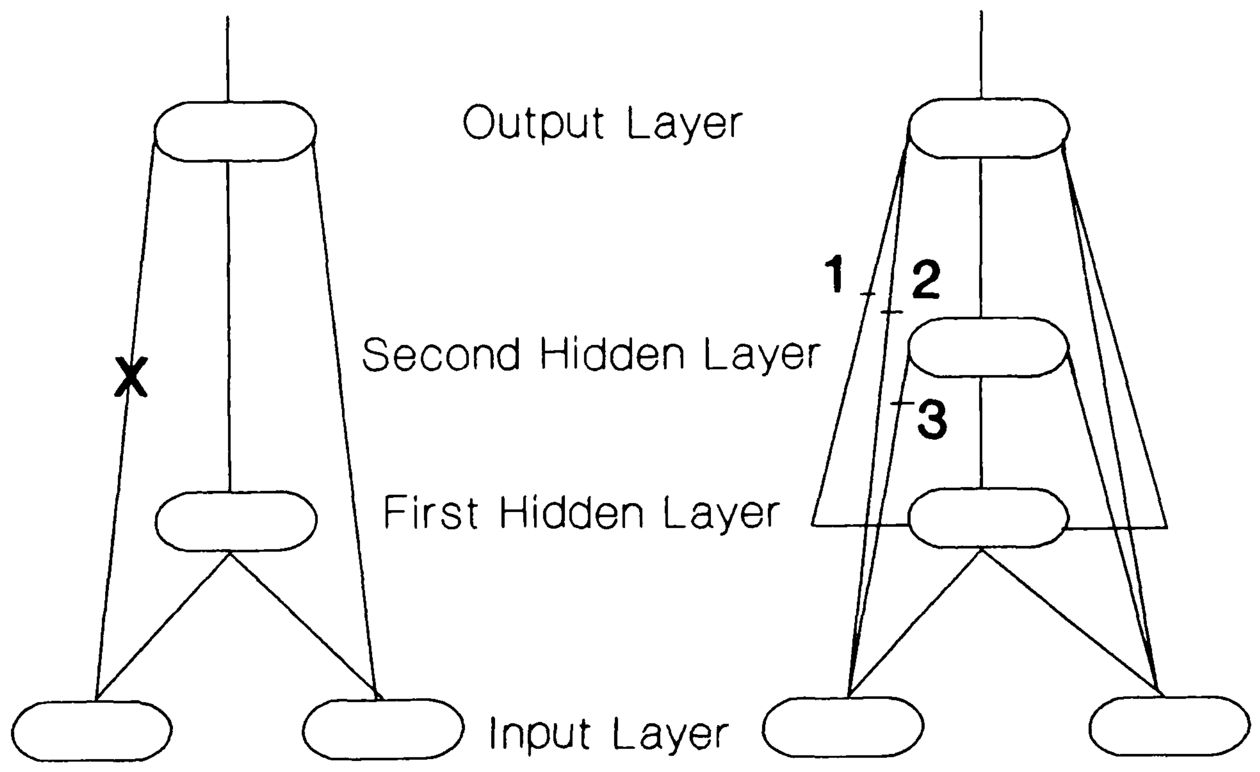
The experiments introduced in the previous section will now be presented in detail.

Three-layer backpropagation:

The three-layer backpropagation model is shown in Fig 1a.

BACKPROPAGATION NEURAL NETWORKS

Fig 1a: Three Layer Network **Fig 1b: Four Layer Network**

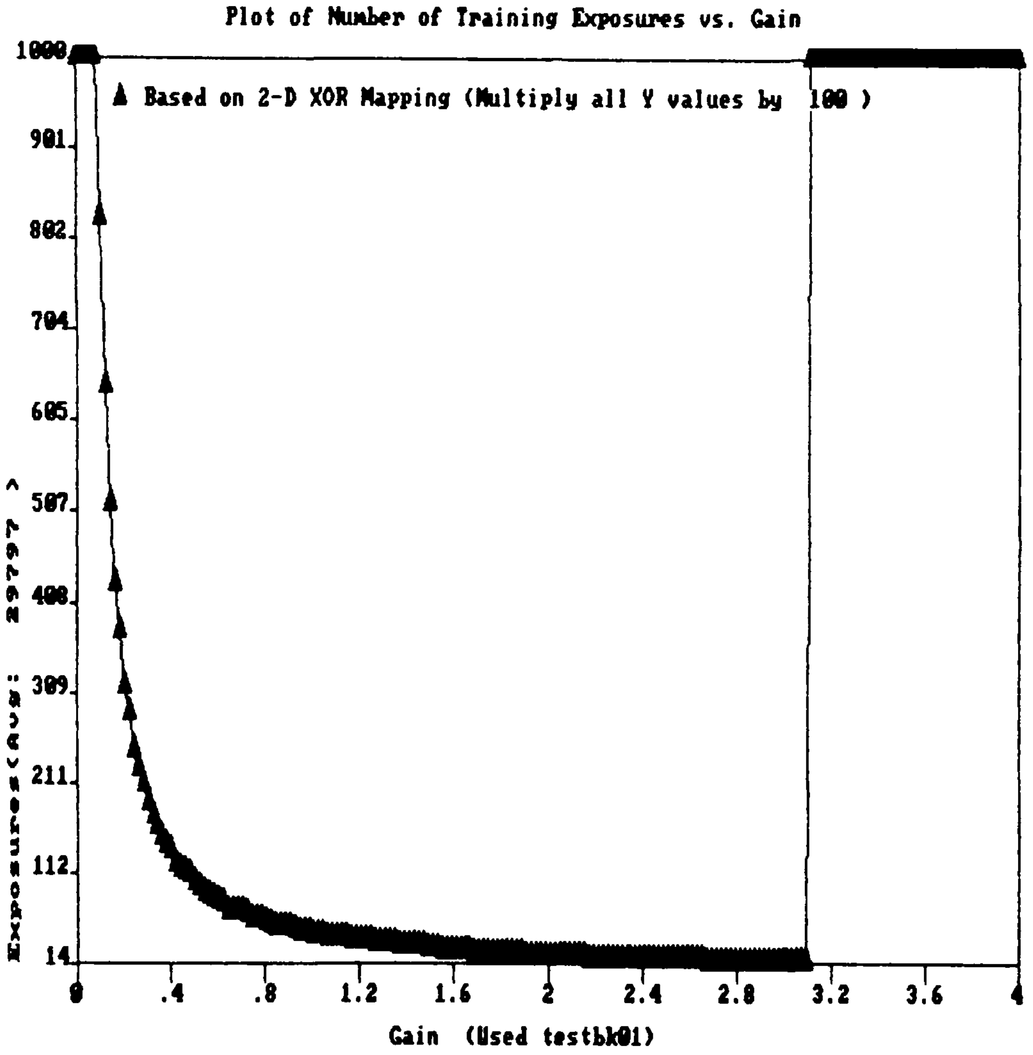


Five experiments with the three-layer backpropagation model were tried:

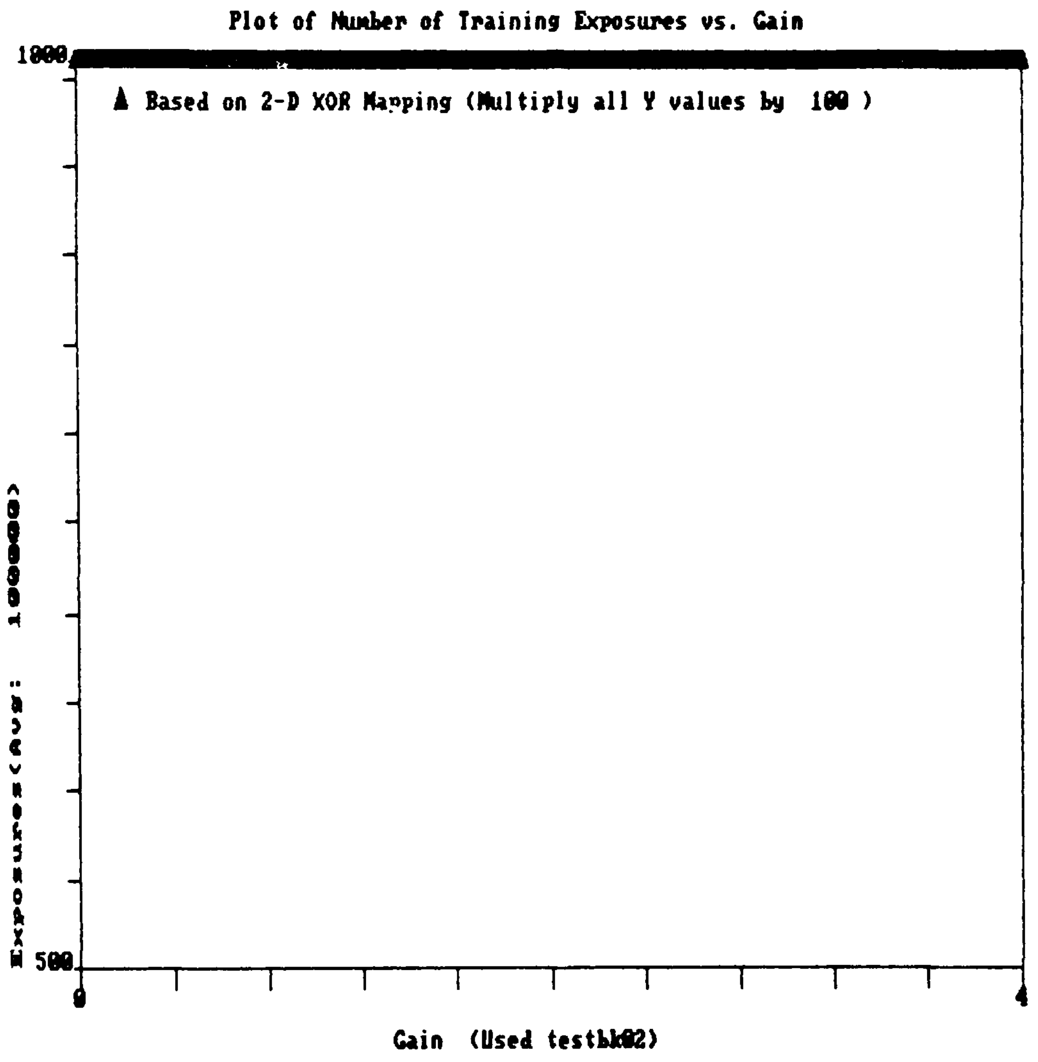
1) data lines marked "X" connected and Gain varied from 0 - 4 in 200 steps
2) data lines marked "X" disconnected and Gain varied from 0 - 4 in 200 steps
3) data lines marked "X" connected, Gain varied from 0 - 4 in 50 steps, and Random Distribution for Initialization varied from U(-.1,.1) to U(-2,2) in 50 steps for each step of Gain.
4) data lines marked "X" connected, Gain varied from 0 - 4 in 50 steps, and Number of Hidden Layer Nodes varied from 1 - 10 in 10 steps for each step of Gain.
5) data lines marked "X" connected, Gain varied from 0 - 4 in 50 steps, and Momentum varied from 0 - 1 in 50 steps for each step of Gain.

Other facts about the runs are: Initialization: U(-.1,.1) for all but Experiment #3; Training: random examples from a two-dimensional XOR table; Computer: DEC MicroVax III under Ultrix using Berkeley Pascal; Weight Updates: Asynchronous within layers, Synchronous between layers; Momentum: 0 except in Experiment #5; Number of Nodes in the Hidden Layer: 1; Acceptable Error: 0.1 for each input/output pair. The random seed was the same for all runs and the generator was reseeded with the original seed after the initial weights were selected.

Experiment 1: No training took place until Gain = .2, as indicated by Fig 2. At that point, training was very slow until Gain = 0.5, at which point training improved to Exposures = 1400 at Gain = 3.2, after which training was again inhibited. We did not try to go beyond Gain = 4 because of the limited computer capacity and because acceptable training had already taken place by then. Exposures refers to the number of randomly selected examples of the input/output mappings the network had to be "exposed to" before it converged.



Experiment 2: No training at all took place. The experiment was terminated at Exposures = 100000 as an upper cutoff at which it was assumed that no training would ever occur. This turned out to be an acceptable limit since other network configurations converged in much less time. Fig 3 illustrates the results of this experiment.



Experiment 3: The point of convergence surfaces is the ability to see where the regions of reliable convergence are. This experiment achieved its fastest convergence time at Gain = 3.10 and Distribution = U(-.41,.41). These parameters are near the outer edge of the low flat region near the right wall. The conclusion is that these values are too near the wall for convergence to be insensitve to small changes in their value. One would be better advised to use something like Gain = 2 and Distribution = (-.5,.5) when trying a new problem in this problem class. These values put one near the middle of the low flat plain. Fig 4 illustrates the results of this experiment. (Unfortunately, the graphics package we used does not give scale numbering. Hopefully, we will be able to get a better package for the next phase.)

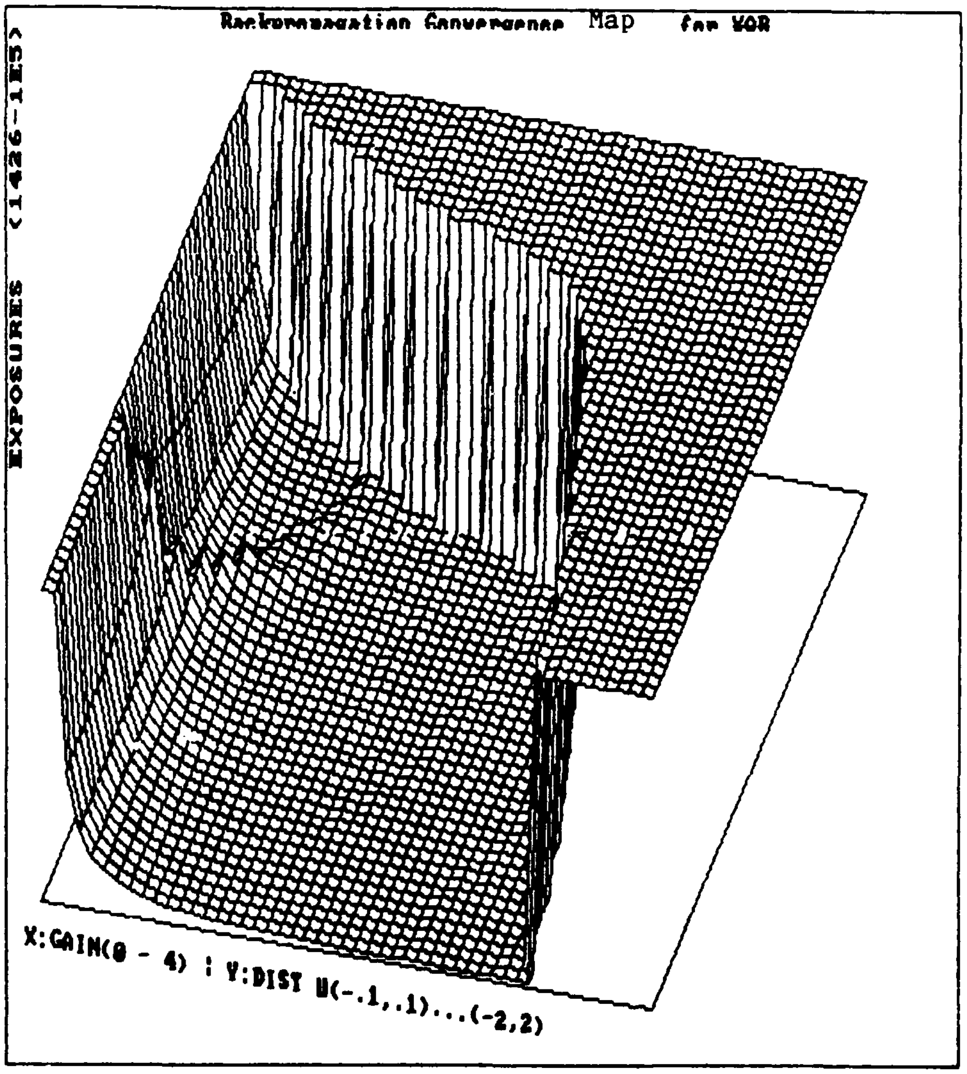


Figure 4. Backpropagation Convergence Map Using Gain and Distribution

Experiment 4: The fastest convergence occured at Gain = 3.51 and Number of Hidden-Layer Nodes = 10. However, note that after Gain = 2.4, variations in Number of Hidden-Layer Nodes causes considerable unreliable network behavior. Thus, there would be no predicting what would happen if a new problem were tried. It would be better to try something like Gain = 1.6 and Number of Hidden-Layer Nodes = 2 in order to get into the low flat region. Fig 5 illustrates the results of this experiment.

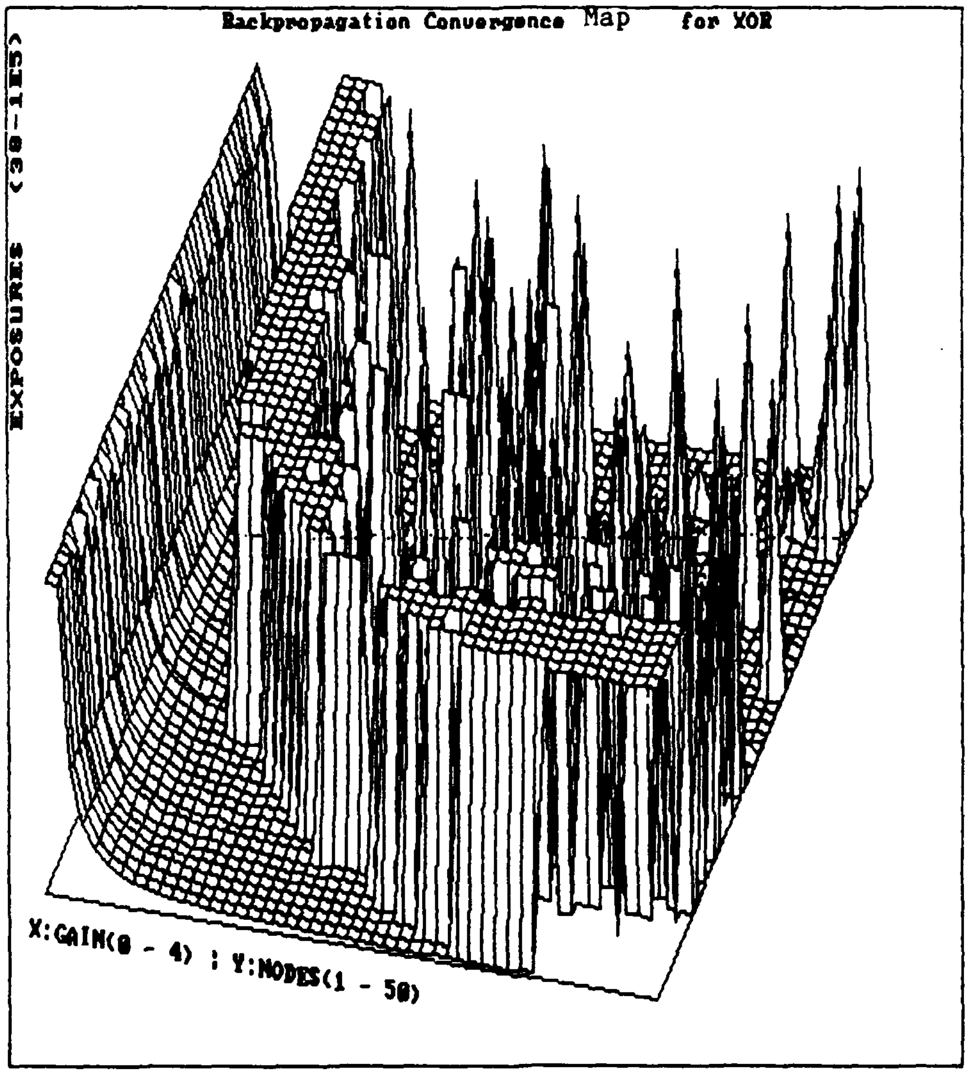


Figure 5. Backpropagation Convergence Map Using Gain and Number of Hidden Nodes

Experiment 5: Convergence was fastest at Gain = 2.53 and Momentum = .63. These values are due to the pit which occurs in the middle of the high flat region. Again, One should not choose these values when trying some new problem. Rather, choose Gain = 1 (or so) and Momentum not greater that .5. Figure 6 illustrates the results of this experiment.

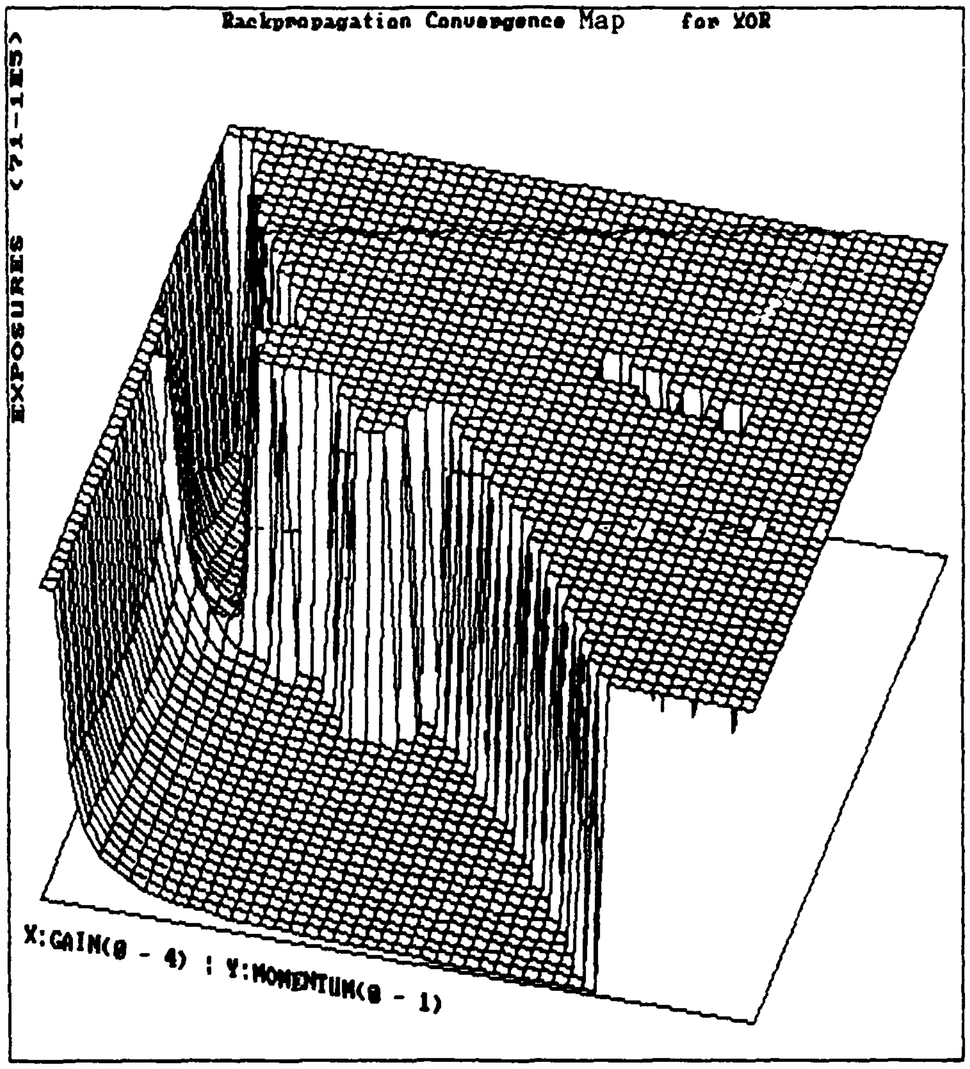


Figure 6. Backpropagation Convergence Map Using Gain and Momentum

Conclusions: From these experiments it is clear that convergence maps can not be used to find the fastest values for training equation parameters and network configurations. This is because the fastest values typically occur in regions where small shifts in value can cause large changes in network performance. It is better to choose values that are in the middle of a low flat region. These values will give slower performance for the training metric but offer more reliable performance for new problems in the same class as the training metric.

Four-layer Backpropagation:

The experiments performed above for three-layer backpropagation neural networks were also performed for four-layer networks. The curves and surfaces, of course, show that the four-layer nets perform differently. In some cases, they perform better.

The following paragraphs report on 14 experiments. Eight experiments developed 2-dimensional maps and the rest developed 3-dimensional maps. The four-layer network is illustrated in Figure 1b. The 2-dimensional maps were developed by varying the network connection architecture and then varying the values of the Gain parameter. Figure 7 charts all variations in connection architecture considered in these experiments.

FIG 7. TWO-LAYER BACKPROPAGATION LINE CLIPPING POSSIBILITIES

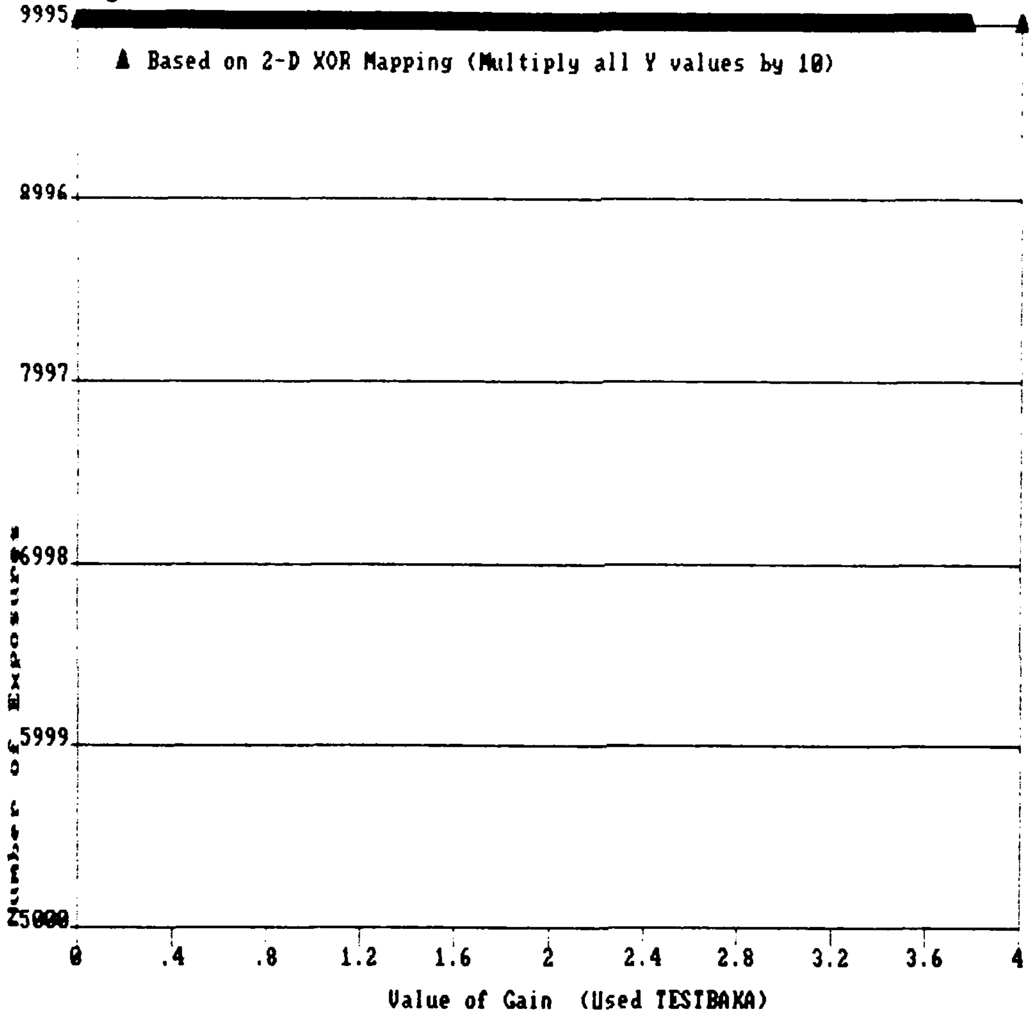
Filename	Connection Scheme		
	Line 1	Line 2	Line 3
Testbaka	Off	Off	Off
Testbakb	Off	Off	On
Testbakc	Off	On	Off
Testbakd	Off	On	On
Testbake	On	Off	Off
Testbakf	On	Off	On
Testbakg	On	On	Off
Testbakh	On	On	On

The remaining experiments produce 3-dimensional maps. One set of 3-dimensional maps was done for the fully connected four-layer network (Experiments 9, 10, & 11) and another for the case where the input nodes were not connected to the output nodes (Experiments 12, 13, & 14).

2-Dimensional Maps: In these experiments, the eight layer-connection architectures given in Figure 7 were used, one at a time, to solve the 2-dimensional XOR problem. For each experiment Gain was incremented from 0 - 4 in 200 trials. The result of each trial then became a point on the plot. Other facts about the runs are: Initialization: U(-.1,.1); Training: random examples from a two-dimensional XOR table; Computer: DEC MicroVax III under Ultrix using Berkeley Pascal; Weight Updates: Asynchronous within each layer, Synchronous between layers; Momentum: 0; Number of Nodes in the Hidden Layers: 1; Acceptable Error: 0.1 for each input/output pair. The random seed was the same for all runs and the generator was reseeded with the same seed after the initial weights were selected.

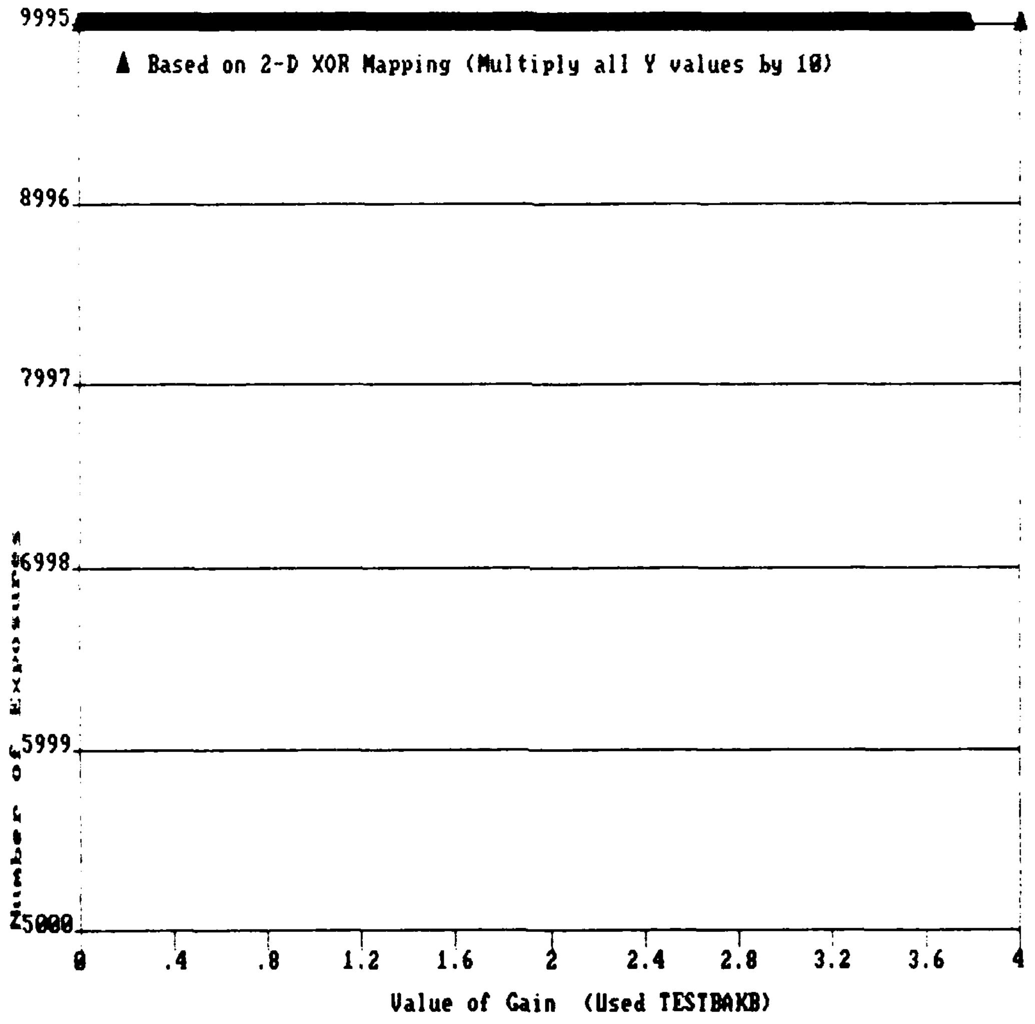
Experiment 1: Using architecture (a) (see Filename = Testbaka in Figure 7) no training at all took place. This was not suprising since the three-layer network which had no lines bypassing hidden layers also did not train at all. The results of this experiment are illustrated in Figure 8.

Figure 8. Plot of Gain vs. Number of Training Exposures



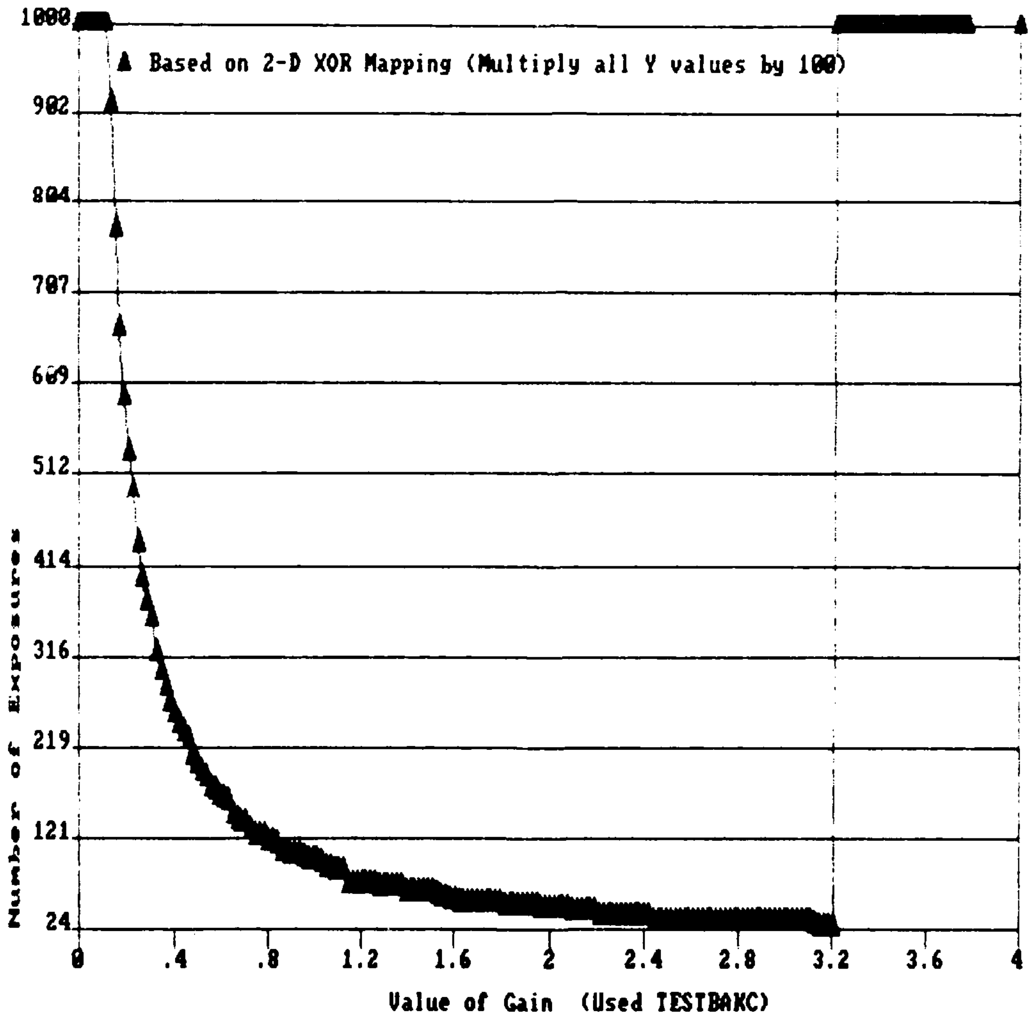
Experiment 2: Using architecture (b) no training at all took place. This was somewhat of a suprise since we had originally expected that training would take place if at least one hidden layer had a bypassing line. As it turns out, connecting a line from the input layer to only the second hidden layer was insufficient to cause training. The results of this experiment are shown in Figure 9.

Figure 9. Plot of Gain vs. Number of Training Exposures



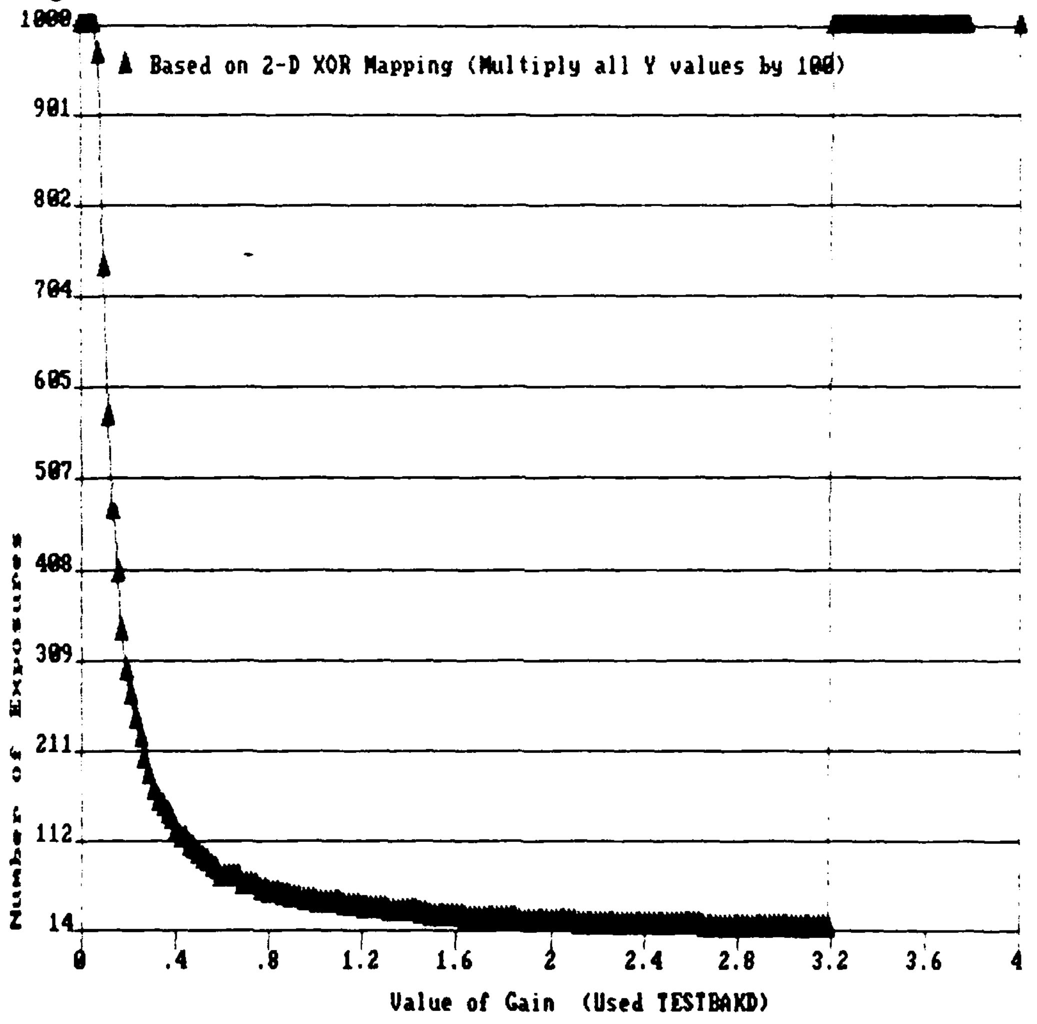
Experiment 3: Using architecture (c) training did take place. In this case a line from the input layer directly to the output layer was sufficient. This seems logical since the same comparable architecture was sufficient for training the three-layer network. Training did not begin until Gain = .3 after which it was slow until Gain = 1.1. Convergence time gradually decreased until Gain = 3.2 after which training was again inhibited. The big difference between the results for this run and the one with three layers is the lesser slope of the four-layer curve. From this one can surmise that if bypass lines are only going to be put in between the input layer and the output layer then the three-layer network will have the better performance for this class of problem. The results of this experiment are shown in Figure 10.

Figure 10. Plot of Gain vs. Number of Training Exposures



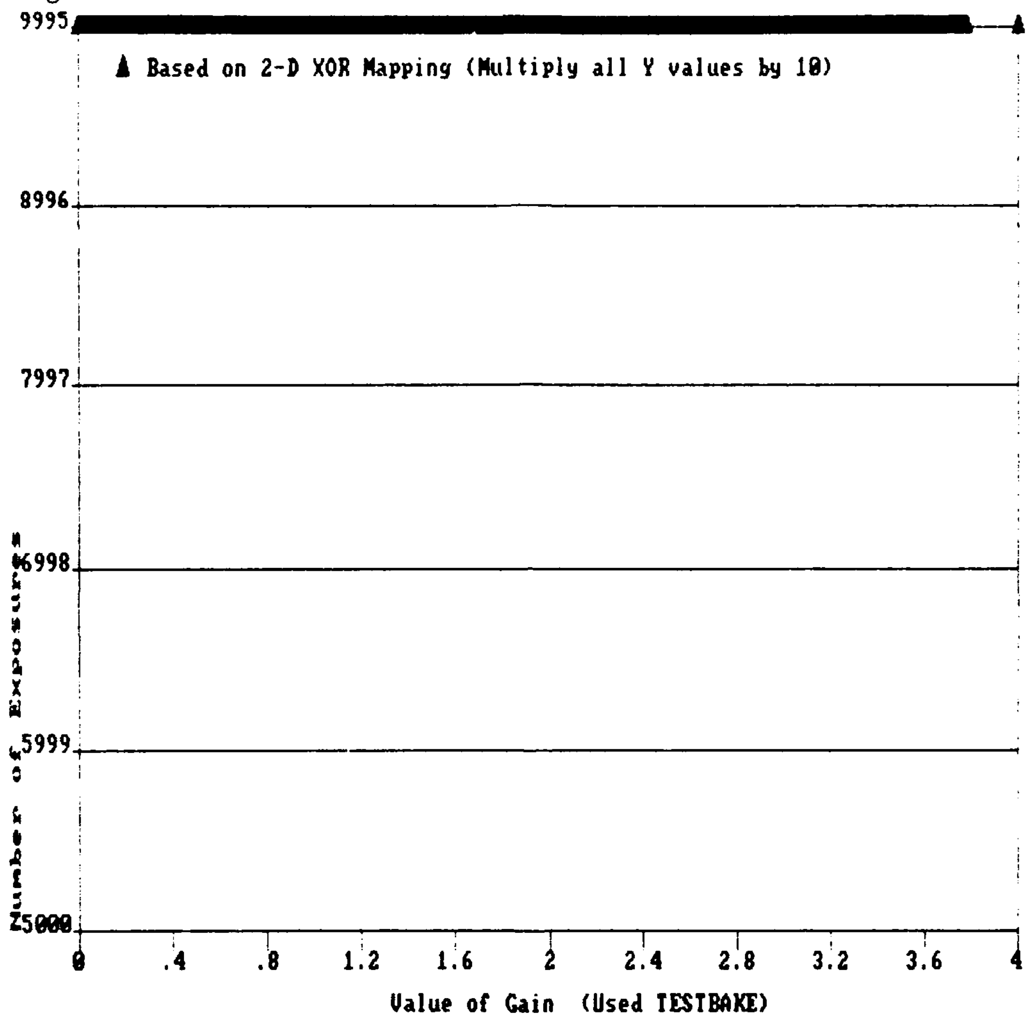
Experiment 4: Using architecture (d) results almost exactly like the three-layer fully connected model were achieved. In architecture (d) lines connecting the input layer directly to the output layer and the second hidden layer were installed. This experiment's results are illustrated in Figure 11. A thought which occurs here is that if the results between two networks are similar then it is better to use the less expensive network. Expense increases as the fan-in to a given layer of nodes increases, as the number of nodes increases, and as the number of layers of nodes increases.

Figure 11. Plot of Gain vs. Number of Training Exposures



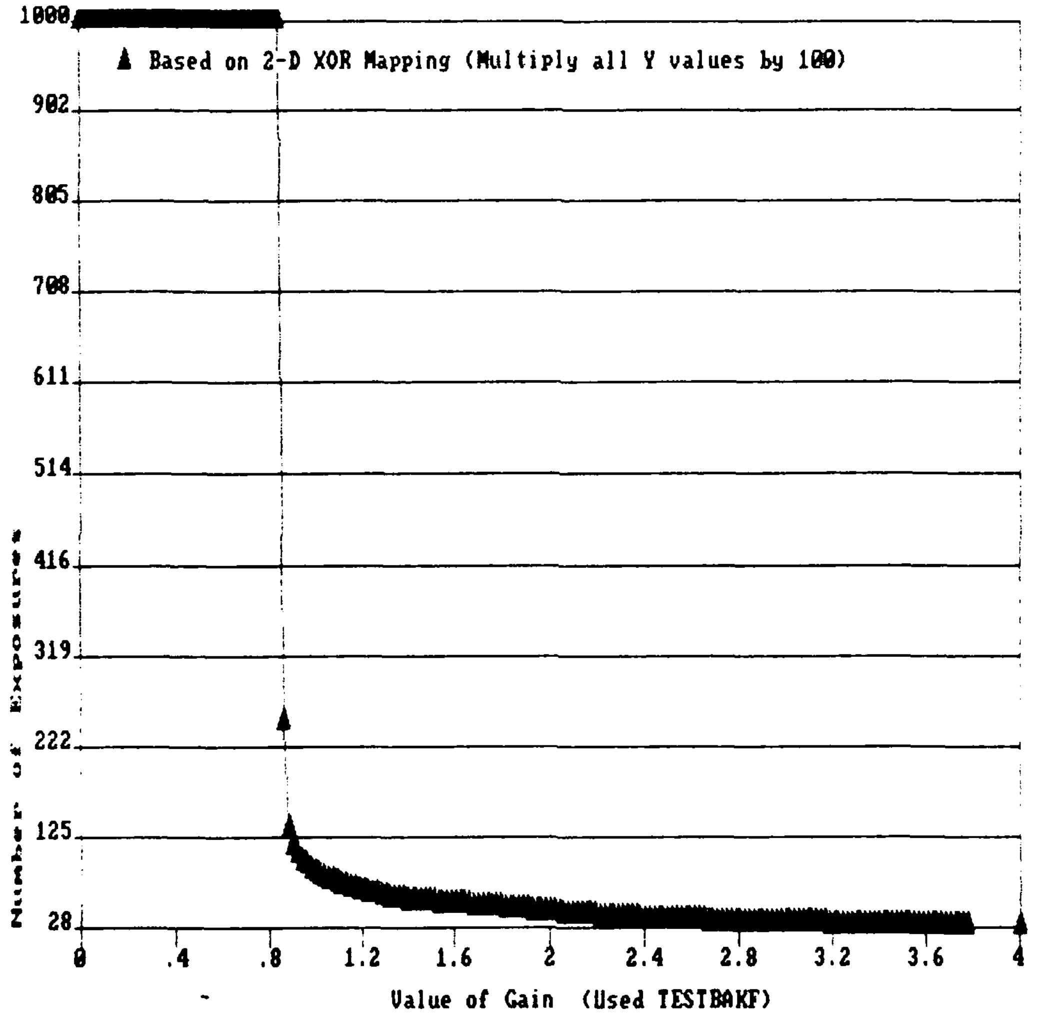
Experiment 5: Using architecture (e) no training at all took place. AGain we see a case where installing only one set of bypass lines in a two-layer network was insufficient to permit convergence. The results of this experiment are illustrated in Figure 12.

Figure 12. Plot of Gain vs. Number of Training Exposures



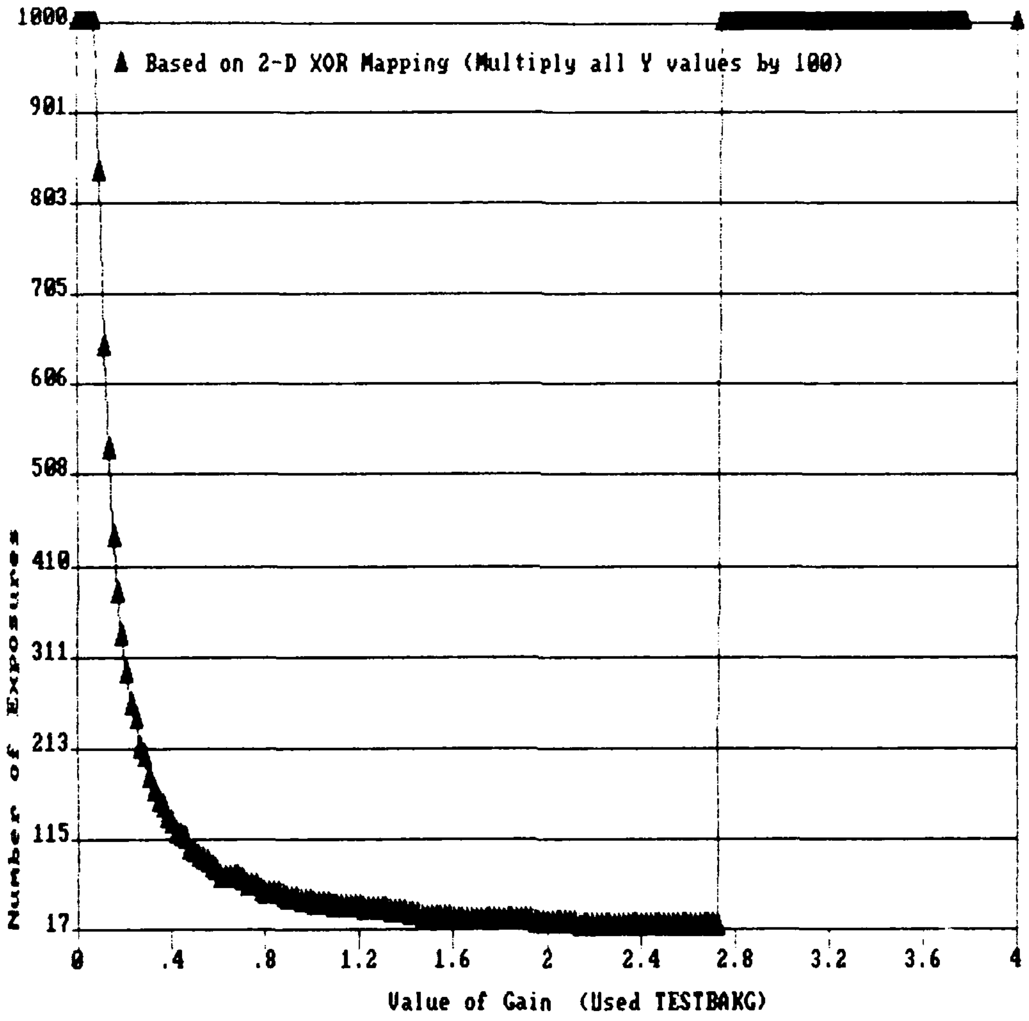
Experiment 6: Using architecture (f) we get an interesting result. There is no training at all until Gain = .85 but there is no point at which training is inhibited after that. The minimum training time is higher than in other experiments but the average training time is lower. It is for this reason we chose this architecture to develop 3-dimensional maps in Experiments 12, 13, and 14. The results of this experiment are illustrated in Figure 13.

Figure 13. Plot of Gain vs. Number of Training Exposures



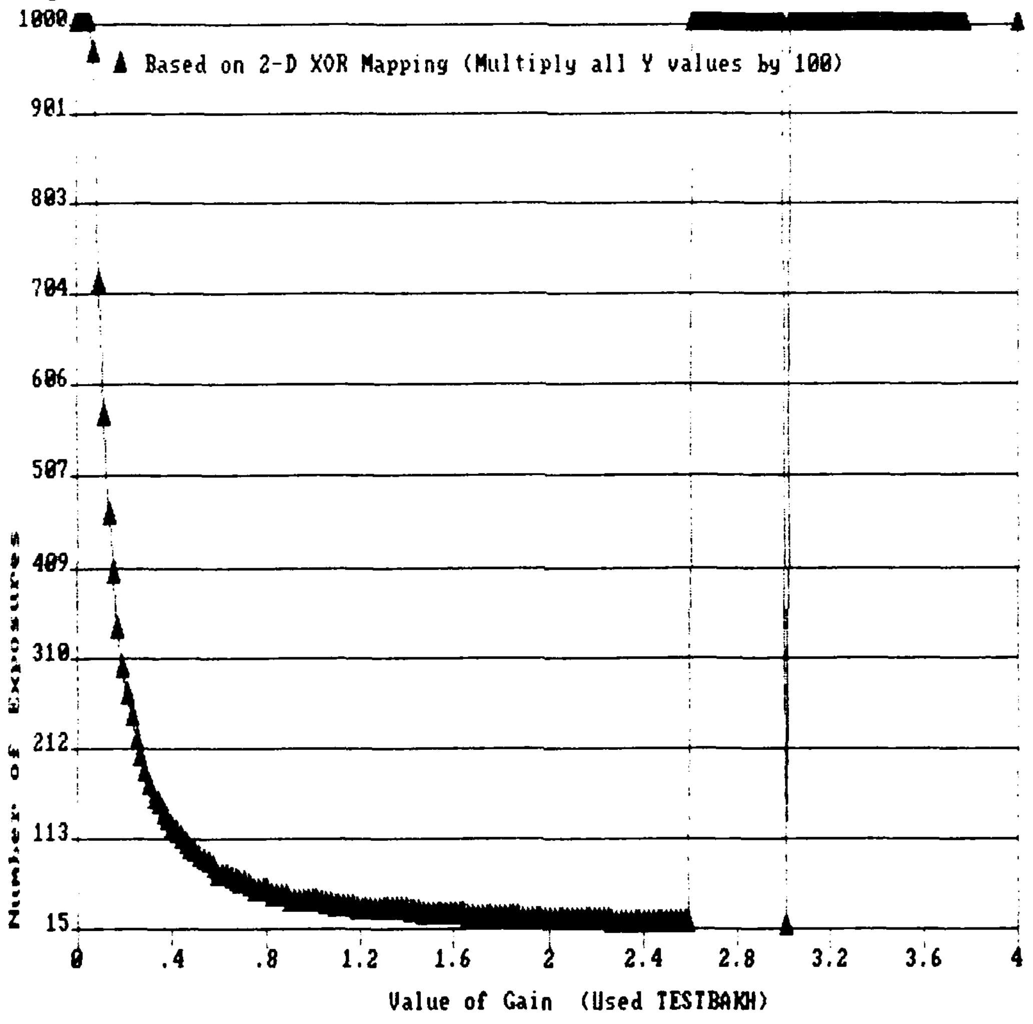
Experiment 7: Using architecture (g) we get a curve that starts off like the three-layer's curve but training gets inhibited at a much earlier time, Gain = 2.75. The results of this experiment are illustrated in Figure 14.

Figure 14. Plot of Gain vs. Number of Training Exposures



Experiment 8: Using architecture (h), the fully connected architecture, we have a curve even worse than that of Experiment 7. Although good training starts early on as in the other experiments where training occured, training is inhibited early too, at Gain = 2.5. In this case too, there is some unreliability which appears at Gain = 3.0. The results of this experiment are illustrated in Figure 15.

Figure 15. Plot of Gain vs. Number of Training Exposures



3-dimensional maps: In these experiments, two different connection architectures were used to solve the 2-dimensional XOR problem. For each experiment, Gain was incremented from 0 to 4 in 50 trials and either Distribution, Number of Hidden Nodes, or Momentum was incremented for each increment of Gain. The result of each trial then became a point on the 3-dimensional surface. Other facts about the runs are: Initialization: U(-.1,.1) except in cases where Distribution was one of the varied parameters; Training: random examples from the two-dimensional XOR table; Computer: DEC MicroVax III under Ultrix using Berkeley Pascal; Weight Updates: Asynchronous within layers, Synchronous between layers; Momentum: 0 except in cases where Momentum was varied; Number of Nodes in Hidden Layers: 1 except in cases where Number of Nodes was varied; Acceptable Error: 0.1 for each input/output pair. The random seed was the same for all runs and the generator was reseeded with the same seed after the initial weights were selected.

The connection architectures used in this experiment were 1) fully connected, input connected to both the output and the hidden layers and 2) input not connected to output. The experiment results given below compare the two architectures' ability to develop the 2-dimensional XOR mappings.

Variations of Momentum: Momentum was varied from 0 to 1 inclusive in 50 steps for each of 50 steps of Gain. Architecture 1): The results with this architecture were similar to that of the fully connected three-layer architecture. As Momentum increases, it has less and less a desirable affect. As Gain increases, Momentum lends less and less assistance to convergence. The fastest convergence occurred at Gain = 3.61 and Momentum = 0.57. However, that is deep in a pit so it is better to choose something like Gain = 1.5 and Momentum = 0.02, values that are in the middle of the low flat plain. The plot for this experiment is shown in Figure 16. Architecture 2): This experiment showed very little opportunity for reliable convergence, only with high values of Gain and low values of Momentum. It would be interesting to extend this plot to Gain = 10, something we may do in the next phase. Convergence was fastest for Gain = 3.51 and Momentum = 0.45. Very near the back-right wall. Figure 17 shows the plot.

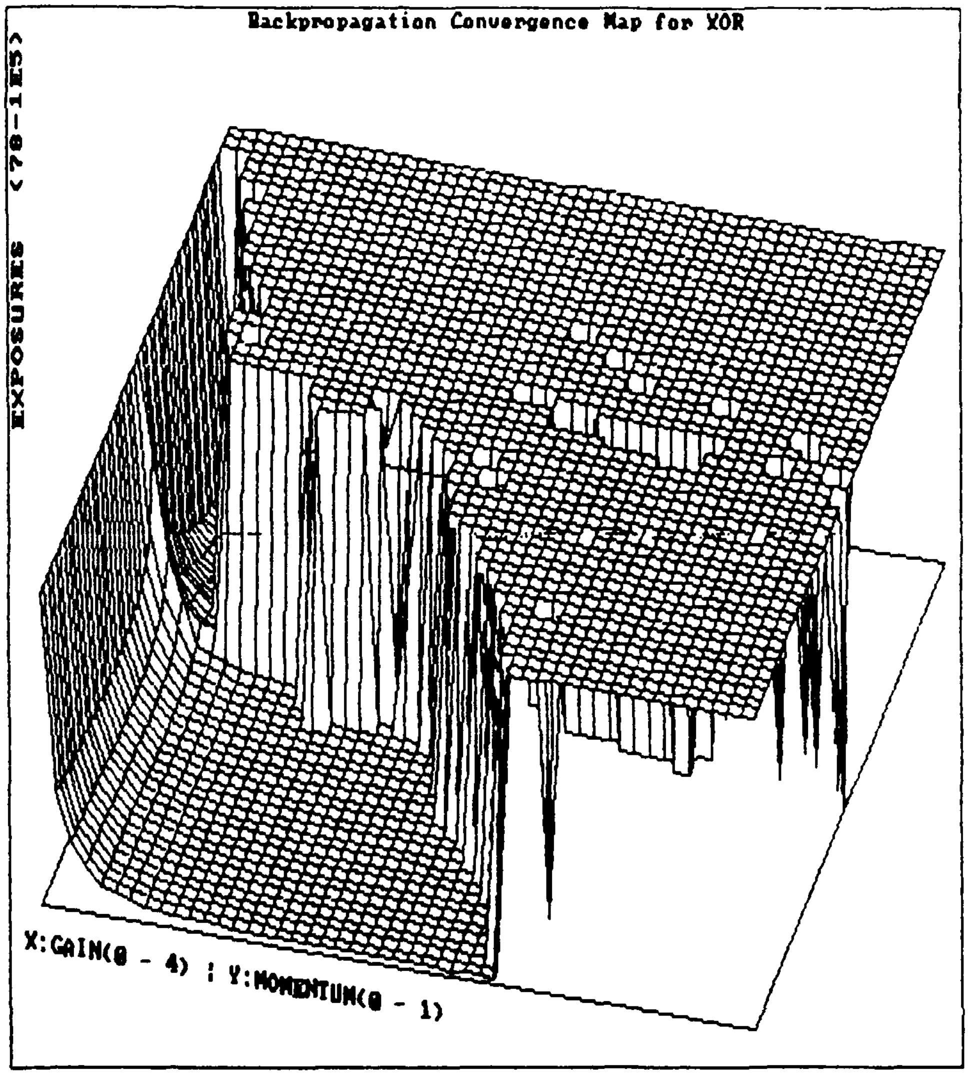


Figure 16. Fully Connected Network Using Gain & Momentum

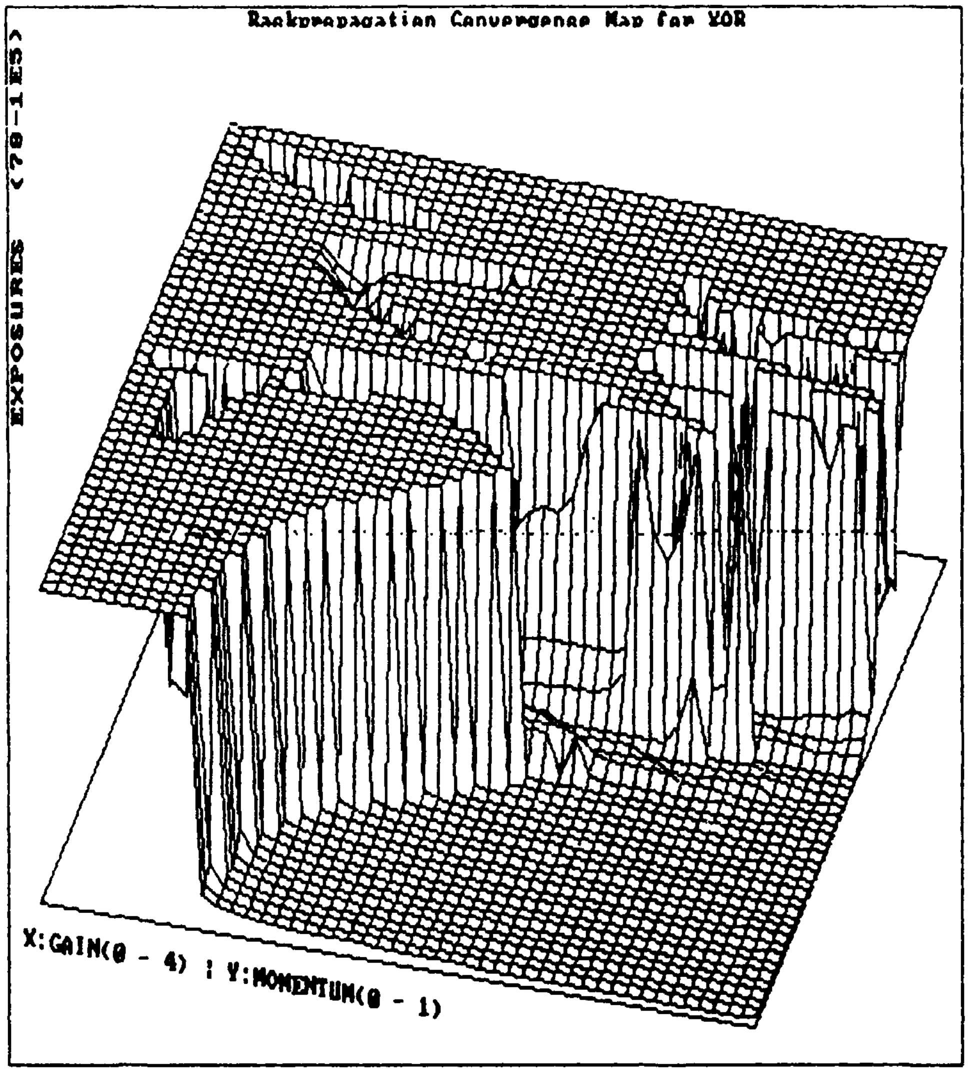


Figure 17. Input not connected to output using Gain & Momentum

Variations of weight initialization distribution: To initialize the network weights, values were chosen randomly from a uniform distribution which varied in 50 steps for each of 50 increments of Gain. The variation was U(-.1,+.1) to U(-2, +2) inclusive. Architecture 1): Distribution variations had very little affect on this architecture's ability to converge. Notice some contrary regions, however. Convergence is difficult for very low values of Gain and very high values of Distribution. Very high values of Gain for most values of Distribution also negatively impact convergence except in the rare case where there is a combination of very high Gain and very large Distribution. Convergence was fastest for Gain = 3.92 and Distribution = 2, in the flat plain which appears in the back-right of the plot. See Figure 18 for this plot. Architecture 2) For this architecture, convergence almost never occurred except for small Distributions. As Gain increased, Distribution generally helped but the affect was minimal. The fastest convergence occurred at Gain = 0.82 and Distribution = 2, a deep pit. Figure 19 illustrates these results.

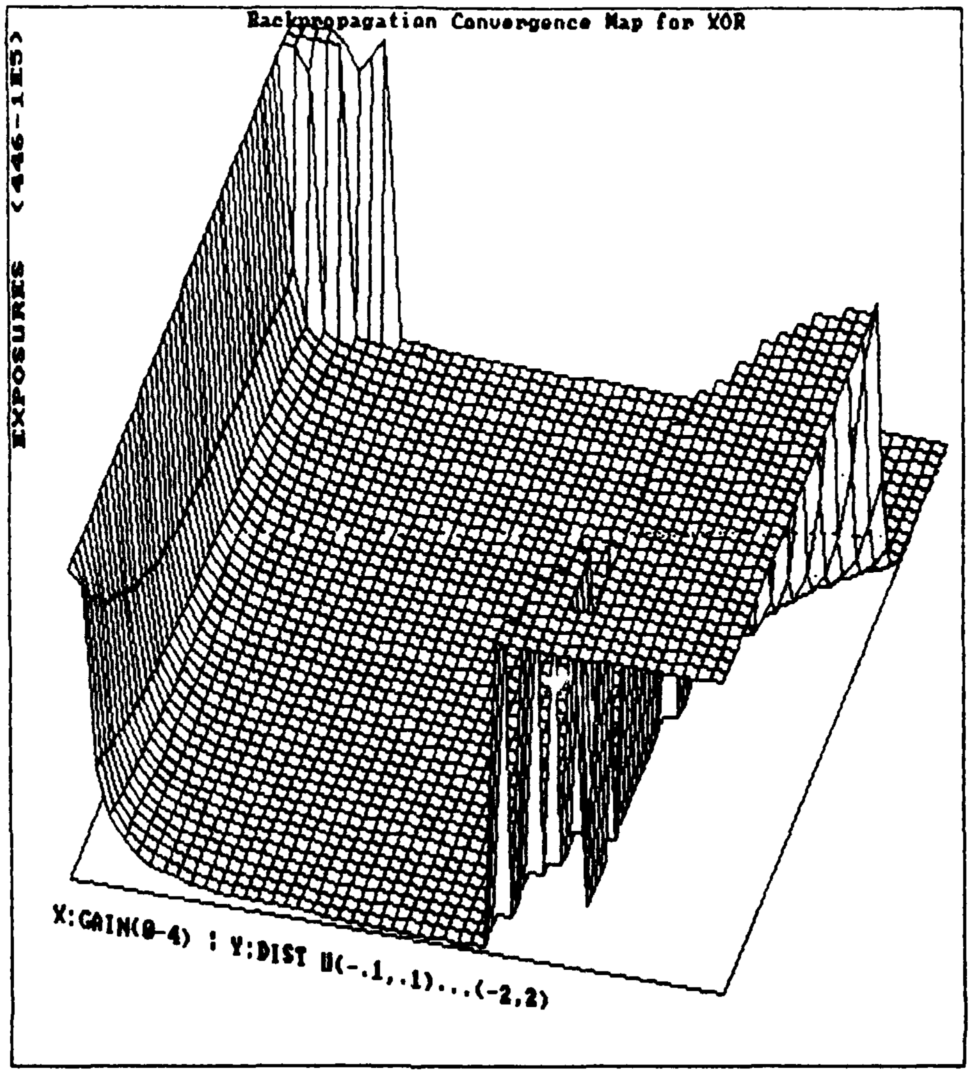


Figure 18. Fully Connected Network Using Gain & Initialization Distribution

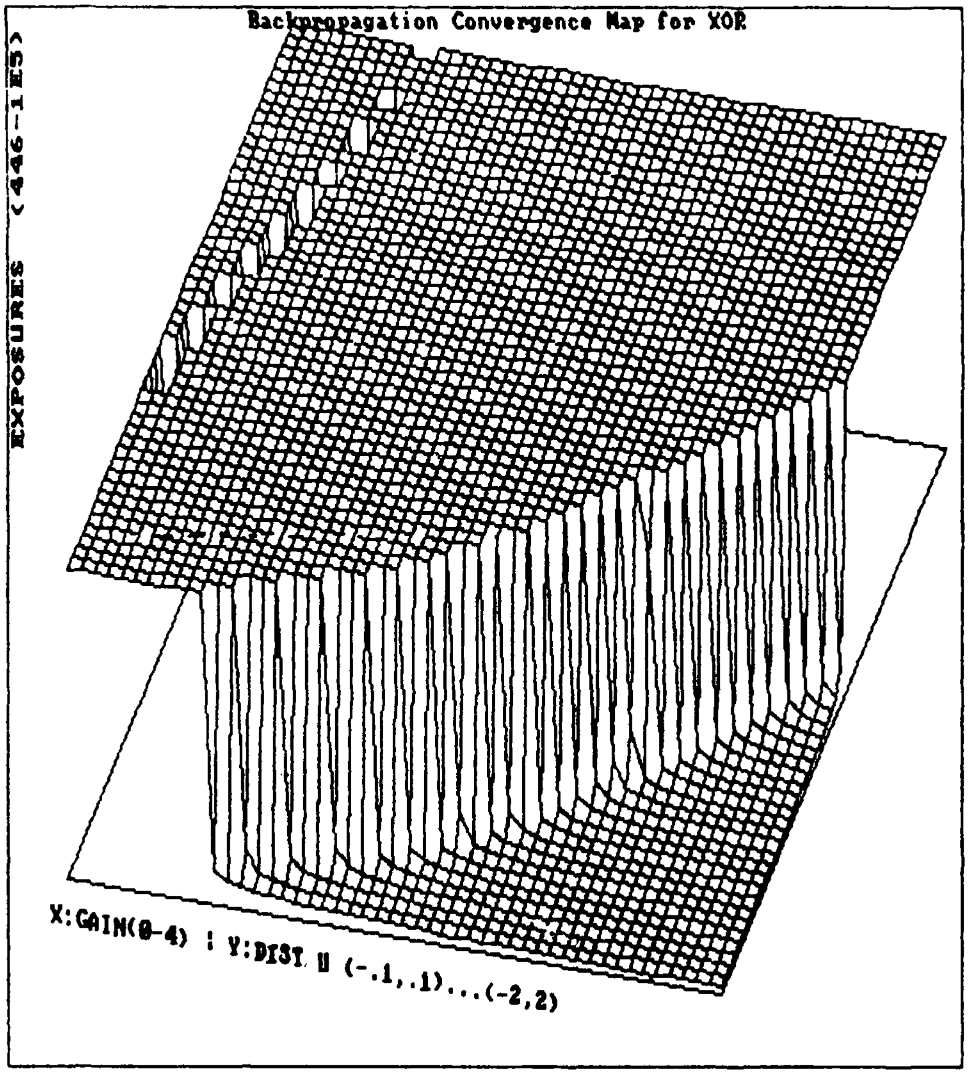


Figure 19. Input not connected to output using Gain & Initialization Distribution

Variations of number of hidden nodes: The number of hidden nodes in both layers was varied from 1 - 10 in 10 steps for each of 50 increments of Gain so that each hidden layer always had the same number of nodes. Only 10 nodes were run since the amount of computer time needed would have been too great to go higher. Architecture 1): Like the fully connected three-layer architecture, variations of number of hidden nodes led to tremendous unreliability in convergence. The fastest time was recorded at Gain = 2.09 and Number of Nodes = 8, in the middle of an unreliable region. See Figure 20 for this plot. Architecture 2) This was a most surprising result in light of the other plots from this architecture. In this case, the second architecture resulted in better performance. The plot is generally like a slide that angles downward, left to right. Only at very low values of Gain is this not so and even then at Nodes = 1. Convergence was fastest at Gain = 3.76 and Nodes = 10. Figure 21 has this plot.

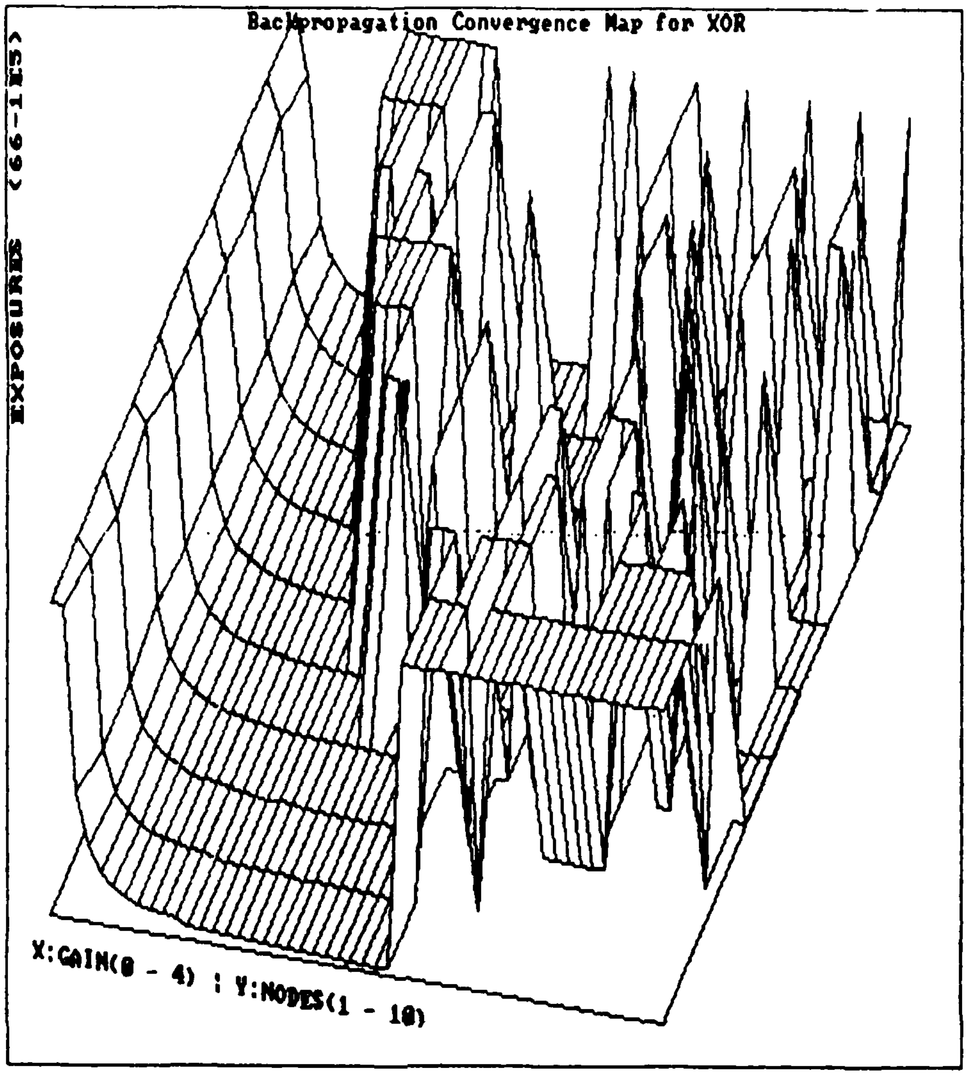


Figure 20. Fully Connected Network Using Gain & Number of Hidden Layer Nodes

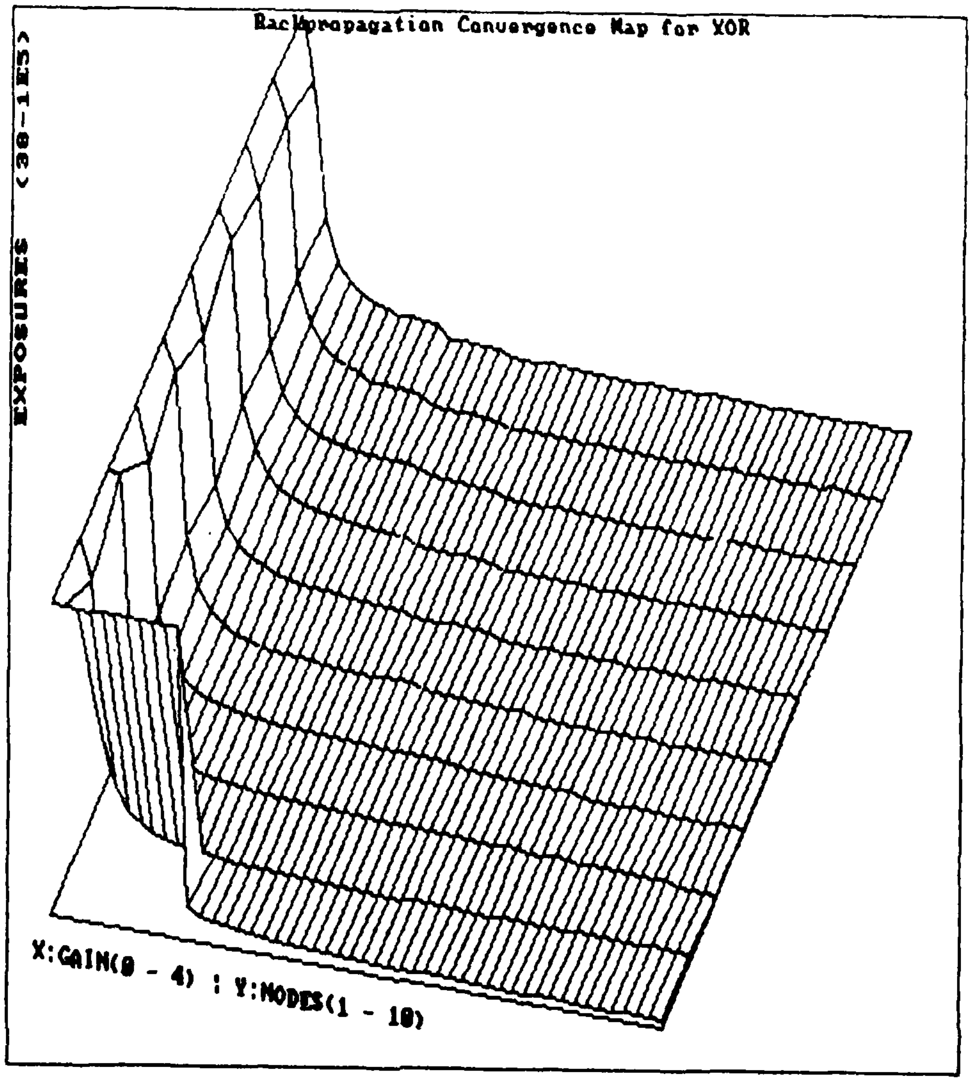


Figure 21. Input not connected to output using Gain & Number of Hidden Layer Nodes

Conclusions

It is obvious from the convergence maps shown in this report that the node connection architecture has a dramatic affect on the network's ability to learn a set of vector mappings. Less dramatic, but just as important, are the values of the network's training equation parameters. Preliminary tests have suggested that these parameters need to be set not to the values which give fastest learning for a given metric but to those values which give reliable learning. Reliable learning ability is shown on the maps as low flat plains. In most of the maps shown in this report, the fastest learning occurred in a pit found on a high flat plain.

A concern which came about as a result of this study has to do with the basic theory of backpropagation. The theory says that backpropagation neural networks are guaranteed to learn an arbitrary set of vector mappings. The theory does not say how long it will take to learn a given set of vector mappings. This is no problem when the data domain is fixed. However, in most military applications, the data domain is not fixed. This leads to questions on what happens if the number of vector mappings to be learned increases and what happens if the number of components in the vectors increases. Certainly, one would expect the number of exposures required for learning to increase. But, what is the rate of increase? Is it linear, geometric, or exponential. It is not sufficient to say, "Well, just put it on your Cray and let it run". In military applications, we have to be able to guarantee reprogrammability within a given amount of time. We will address these issues in the next phase of this project when we begin using the random bit replacement metric.

Description of Convergence Maps

Convergence maps are N-dimensional plots which show the ability of a neural network to converge on (learn) a given training metric. There are several training metrics in common use for testing input/output vector mapping networks: . Among these are: 2-dimensional XOR, 3-dimensional XOR, 4-2-4 encoder, 2-bit adder, contiguity patterns, the prostrate eight, and random-bit-stream replacement. The traveling salesman optimization problem is a classic metric for testing energy minimization networks.

Convergence maps show the performance of a neural network under varying training equation parameters and network configurations. Variations of parameter and configuration are plotted against the time it takes the network to become trained under a given variation. By observing the surface, one can see directly how to set the values of the training equation and the network configuration for a given problem class in order to achieve convergence which is not sensitive to small variations in parameter value or network configuration. A convergence map be plotted for N = 2, 3, or 4 dimensions. The plot is based on value variations of N-1 training equation parameters or network configurations. The Nth dimension is always time-to-convergence.

For instance, in classical backpropagation (see Lippmann), there are two training equation parameters, Gain and Momentum. Gain is related to the speed of training. Momentum is related to the expectation that the result of training at time t+1 will be the same as the result of training at time t.

The convergence map in Figure A1-1 shows how the classical backpropagation neural network behaves when being trained to recognize an XOR table. To get this map, the Gain was varied from 0 - 4 in 50 steps and the Momentum was varied from 0 - 1, each in 50 steps for each value of gain. Training was halted at 100000 trials for any combination of Gain and Momentum that did not recognize the table (converge) in that many trials. The map shows that the best picks for Gain start after about .5 and that Momentum has less and less a desirable affect as Gain increases. The axes for this map, and all maps in this report, are explained in Figure A1-2.

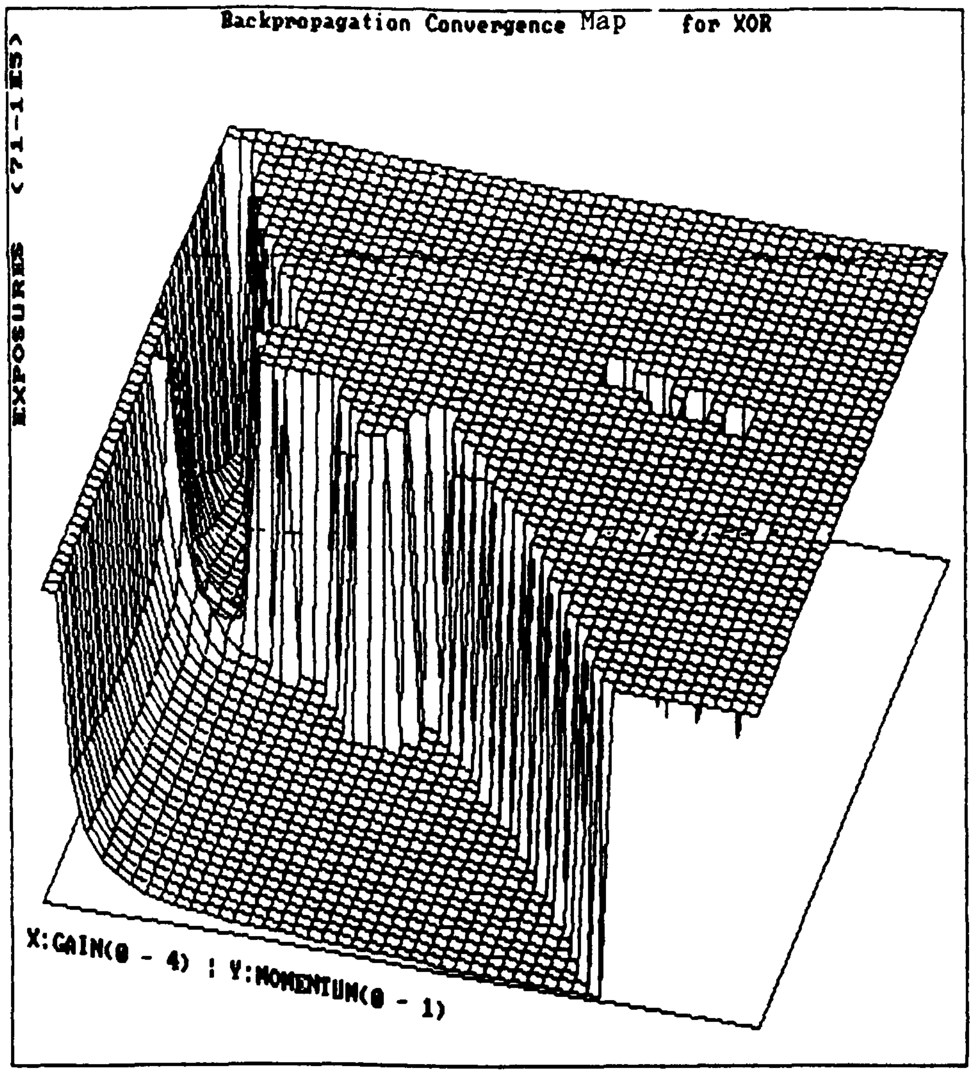


Figure A1-1 Backpropagation Map Surface

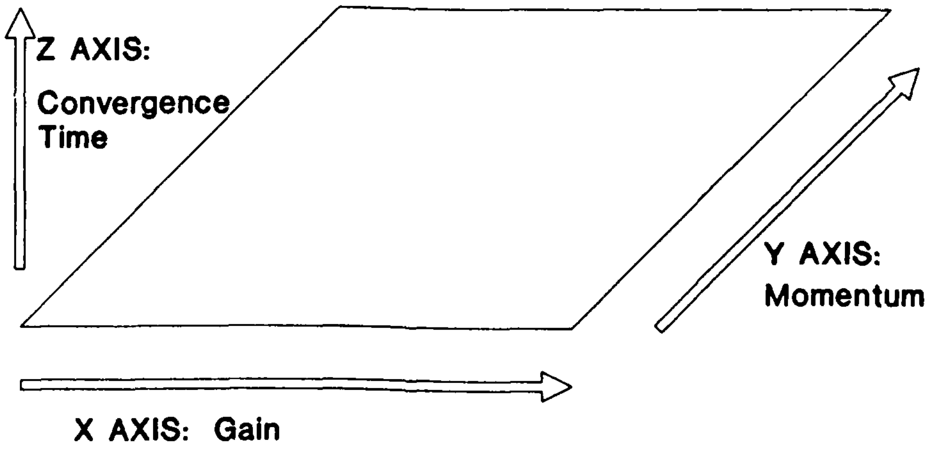


Figure A1-2: Example 3-Dimensional Convergence Map Axes

Pulse-Train Restoration Problem Statement

When a radio frequency (RF) signal is deinterleaved, the pulses thus obtained are assigned to various pulse trains according to the deinterleaver's algorithm. For instance, the deinterleaver may decide that there are two separate pulse trains contained in the received RF signal. The deinterleaver will then assign the pulses it sees to one of the two trains according to the deinterleaving algorithm. The number of trains perceived to be present is not a constant.

Since the receiver system and the deinterleaving process are not perfect, it can happen that pulses are assigned to the wrong chain and that pulses are missed altogether. Thus, it can happen that any given pulse train will have wrongfully added pulses and/or missing pulses. During the threat alert process, it would be useful if the pulse trains could be restored to their original form.

If one assumes that the deinterleaving process is correct most of the time, one could use the past history of any pulse train to predict the future form of the pulse train. The predicted form of the pulse train could be compared to the deinterleaved form of the pulse train. Discrepancies could be fed through an algorithm or set of rules to determine if a given pulse should be removed, a pulse added, or the chain left as is.

An early research goal should be to exceed 50% probability of correct restoration with a confidence level of 99%

Bibliography

Gustafson, Steve "Class Notes on Backpropagation" University of Dayton Research Institute, Jan 89

Klimasauskas, Casimir C. "Neural Networks: A Short Course", PC AI Magazine, Nov/Dec 88

Lippmann, Richard P. "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, Apr 87

Lutey, Mark K. "Problem Specific Applications for Neural Networks", Thesis: Air Force Institute of Technology, AFIT-GE-ENG-88D-23, Available through DTIC/NTIS, Dec 88

Raeth, Peter G. "Backpropagation Neural Network Modeling and Demonstration", Unpublished notes, Available from the author, Jan 89

Raeth, Peter G. "3-D Surface Maps for Neural Network Performance Evaluations", Dayton SIGART Aerospace Applications of Artificial Intelligence Conference, Oct 89